IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of: Dettinger et al. §
Serial No.: 10/720,960 §
Filed: November 24, 2003 §
For: DYNAMIC FUNCTIONAL MODULE AVAILABILITY §

Confirmation No.: 5201
Group Art Unit: 2168
Examiner: Mahesh H. Dwivedi

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450, or facsimile transmitted to the U.S. Patent and Trademark Office to fax number 571-273-8300 to the attention of Examiner Mahesh H. Dwivedi, or electronically transmitted via EFS-Web, on the date shown below:

August 3, 2007 /Randol W. Read, Reg. No. 43,876/
Date Randol W. Read

Dear Sir:

RESPONSE TO NOTICE OF NON-COMPLIANT APPEAL BRIEF

In response to the Notice of Non-Compliant Appeal Brief, Appellants submit this Substitute Appeal Brief to the Board of Patent Appeals and Interferences in response to the Notification of Non-Compliant Appeal Brief dated July 3, 2007 in the Appeal of the above-identified application.

TABLE OF CONTENTS

Real Party in Interest

The present application has been assigned to International Business Machines Corporation, Armonk, New York.

Related Appeals and Interferences

Applicant asserts that no other appeals or interferences are known to the Applicant, the Applicant's legal representative, or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

Status of Claims

Claims 10-12, 14-21 and 27-29 are pending in the application. Claims 1-29 were originally presented in the application. Claims 1-9, 13 and 22-26 have been canceled without prejudice. Claims 10-12, 14-21 and 27-29 stand finally rejected as discussed below. The final rejections of claims 10-12, 14-21 and 27-29 are appealed. The pending claims are shown in the attached Claims Appendix.

Status of Amendments

All claim amendments prior to the Final office Action have been entered by the Examiner. Proposed amendments to the claims after the final rejection were not entered.

Summary of Claimed Subject Matter

A. CLAIMS 10 AND 20 – INDEPENDENT

Claim 10 is directed to a method of providing a user access to functional modules from within an application used to build queries, issue queries, and/or view query results during a query session. The method comprises assigning metadata requirements to functional modules that operate on data stored in, or functional modules that generate results that are stored in, a database, wherein the assigned metadata requirements specify conditions required for successful execution of the functional module. *See* ¶ 42; Figure 1: items 129 and 160. The method also comprises collecting run time metadata relating to a query session, wherein the metadata is collected after the composition of a query. *See* ¶ 47-51; Figure 3A: items 301-304, 170, 172, 174, 176. The method further comprises obtaining a list of functional modules that are accessible from within the application used during the query session. *See* ¶ 51-52; Figure 3A: items 305 and 129; Figure 3B: item 352. Additionally, the method comprises identifying a limited subset of the functional modules in the list that will successfully execute, by comparing the collected runtime metadata with the assigned metadata requirements. *See* ¶ 51-53; Figure 3A: items 306 and 129_B; Figure 3B: items 353-358. The method also comprises providing an interface presenting the user with the identified limited subset of functional modules that will successfully execute. See ¶ 57; Figure 4C: item 430_C.

Claim 20 is directed to a computer readable medium containing a program which, when executed, performs the method described in claim 10. *See* ¶ 41-42; Figure 1.

B. CLAIM 27 – INDEPENDENT

Claim 27 is directed to a data processing system for providing a user access to functional modules from within an application. The data processing system generally comprises a data repository and a plurality of functional modules, each having associated metadata requirements that specify conditions required for successful execution of the functional modules. *See* ¶ 42; Figure 1: items 156, 129 and 160. The data processing system further comprises an application from which the functional

modules are accessible, wherein the application is configured to collect runtime metadata after the composition of a query and present to a user a limited subset of the functional modules that will successfully execute, as determined by the application based on the collected runtime metadata and the metadata requirements associated with the functional modules. ¶ 47-53; Figures 3A, 3B, 4C.

Grounds of Rejection to be Reviewed on Appeal

1. Rejection of claims 10-12, 14-21, and 27-29 under 35 U.S.C. 103(a) as being unpatentable over *Win et al.* (U.S. Pat. No. 6,453,353, hereinafter, "*Win*") and in view of *Pazandak et al.* (U.S. Pat. No. 7,027,975, hereinafter, "*Pazandak*").

ARGUMENTS

Rejection of Claims 10-12, 14-21 and 27-29 under 35 U.S.C. § 103(a) over *Win* in view of *Pazandak*.

The Applicable Law

The Examiner bears the initial burden of establishing a *prima facie* case of obviousness. *See* MPEP § 2142. To establish a *prima facie* case of obviousness three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one ordinary skill in the art to modify the reference or to combine the reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *See* MPEP § 2143. The present rejection fails to establish at least the third criterion.

The References

Win is directed to a method for controlling access to one or more web resources stored on a web server. The method in *Win* comprises defining a "role" for each user of a given system. *See Win*, Column 5: Lines 21-23. A role reflects a relationship of a user to an organization (e.g. employee, customer, distributor, supplier), to a department within an organization (e.g. sales, marketing, engineering), or any other function or affiliation that defines their information needs and thus their access rights or privileges. *See Win*, Column 5: Lines 23-29. Roles determine what resources a user can access. *See Win*, Column 5: Lines 44-46. Therefore, the user is presented only with an authorized list of resources after logging on to the system. *See Win*, Column 6: Lines 10-14.

Pazandak is directed to a method for a guided natural language interface. The method comprises inputting a query to a client and communicating to an interface intermediary. *See Pazandak*, Column 3: Lines 58-59; Figure 4. The method further comprises communicating with an interface descriptor data source, generating an

interface descriptor, communicating the interface descriptor to the interface intermediary, and communicating the interface descriptor to a parser farm. *See Pazandak*, Column 3: Lines 59-63; Figure 4. The parser farm may identify an appropriate parser. *See Pazandak*, Column 10: Lines 50-56; Figure 4. The parser may present a user with next choices of the language for the query to receive further input of the query, and parse the query input. *See Pazandak*, Column 10: Lines 63-67; Column 11: Lines 1-4; Figure 4. The entire parsed and translated query may then be executed by a target application. *See Pazandak*, Column 11: Lines 5-20; Figure 4.

Applicants' Argument

I. *Win* Does Not Disclose Assigned Metadata Requirements That Specify Conditions For Successful Execution of a Functional Module.

Examiner interprets "assigned metadata requirements" to be analogous to the "roles" described in *Win*. See page 4 of Final Office Action dated November 1st, 2006. Examiner also interprets "functional modules" to be analogous to the resources described in *Win*. *See Id*. However, *Win* does not disclose assigned metadata requirements that specify conditions for successful execution of a functional module. In other words, the roles described in *Win* do not *define conditions for successful execution of a resource*, as do the assigned metadata requirements recited in the claims. The roles simply "determine what resources a User can access." *See Win*, Column 5: Line 44.

A resource is defined by *Win* as "a source of information, identified by a Uniform Resource Locator (URL) and published by a Web server either in a static file formatted using Hypertext Markup Language (HTML) or in a dynamically generated page created by a CGI-based program." *See Win*, Column 5: Lines 12-19. "Examples of resources include a Web page, a complete Web site, a Web-enabled database, and an applet." *See Win*, Column 5: Lines 19-20. A role simply limits the resources to a unique set of resources of the available resources based on the particular role of the user. *See Win*, Column 5: Lines 44-54; Column 6: Lines 10-16. However, a role does not specify conditions for the successful execution of a particular resource.

Furthermore, as applicants point out in the Response to Final Office Action dated January 3rd, 2007, "successful execution of a functional module" is not the same as limiting a set of available resources based on a user's role. While a role may present a user with a limited set of resources, a role, as defined in *Win*, does not ensure successful execution of a resource. For example, a resource made available by a user's role may require three input parameters to generate a result, even when executed by a user with full administrative privileges. However, if only a single input parameter is provided, the resource will not successfully execute, regardless of the user's role. In contrast, the metadata requirements as recited in the claims *specify conditions for successful execution of a functional module*. See the last three sentences of ¶ 44.

Accordingly, *Win* does not disclose assigned metadata requirements that specify conditions for successful execution of a functional module.

II. The Combination of *Win* and *Pazandak* Does Not Disclose Collecting Runtime Metadata Relating to a Query Session After the Composition of the Query.

As discussed in the previous section, the Examiner interprets "user roles" in *Win* to be analogous to "runtime metadata." Examiner acknowledges, however, that *Win* does not teach collecting runtime metadata relating to a query session, wherein the metadata is collected after the composition of a query because *Win's* "roles" are not "collected after the composition of a query" (*Win* column 5, lines 60-65). Examiner therefore relies on *Pazandak*, wherein a parser responds to an end-user's selection from menu choices by returning a list of next choices (*Pazandak* column 12, lines 59-62). Examiner notes, the parser could send "the set or subset of the Interface Descriptor 306, *e.g.*, LL Parser ID, transaction ID, or other metadata" (*Pazandak* column 17, lines 16-20). Therefore, Examiner suggests that *Pazandak* teaches collecting runtime metadata relating to a query session, wherein the metadata is collected after the composition of a query.

However, to rely on *Pazandak* "runtime metadata" must be interpreted as "End-User selection from the menu choices," and not as "user roles." Under an interpretation

of "runtime metadata" as "End-User selection from the menu choices," the combination of prior art references fail to teach identifying a limited subset of the functional modules in the list that will successfully execute, by comparing the collected runtime metadata with the assigned metadata requirements, as recited in the claims.

Were the two references combined, the "collected runtime metadata" could not be both "user roles" and "End-User selection from the menu choices." If the "collected runtime metadata" comprised "user roles," then the combination of the references would fail to teach *collecting runtime metadata relating to a query session, wherein the metadata is collected after the composition of a query*. If the "collected runtime metadata" comprised "End-User selection from the menu choices," then the combinations of the references would fail to teach *identifying a limited subset of the functional modules in the list that will successfully execute, by comparing the collected runtime metadata with the assigned metadata requirements*. Were it possible to successfully combine the two references, the proposed combination would fail to teach or suggest all of the claim limitations. The present rejection, thereby, fails to establish at least the third criteria for a *prima facie* case of obviousness.

For these reasons, Applicants submit independent claims 10, 20, 27, and the dependents therefrom are believed to be allowable and allowance of the claims is respectfully requested.

CONCLUSION

The Examiner errs in finding that claims 10-12, 14-21 and 27-29 are unpatentable over *Win* in view of *Pazandak* under 35 U.S.C. § 103(a).

Withdrawal of the rejection and allowance of all claims is respectfully requested.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

/Randol W. Read, Reg. No. 43,876/

Randol W. Read
Registration No. 43,876
Patterson & Sheridan, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Appellants

CLAIMS APPENDIX

1-9. (Canceled)

10. (Previously Presented) A method of providing a user access to functional modules from within an application, used to build queries, issue queries and/or, view query results, during a query session comprising:

assigning metadata requirements to functional modules that operate on data stored in, or functional modules that generate results that are stored in, a database, wherein the assigned metadata requirements specify conditions required for successful execution of the functional module;

collecting runtime metadata relating to a query session, wherein the metadata is collected after the composition of a query;

obtaining a list of functional modules that are accessible from within an application used during the query session;

identifying a limited subset of the functional modules in the list that will successfully execute, by comparing the collected runtime metadata with the assigned metadata requirements; and

providing an interface presenting the user with the identified limited subset of functional modules that will successfully execute.

11. (Original) The method of claim 10, wherein the runtime metadata comprises attributes of fields involved in a query or query results.

12. (Original) The method of claim 10, wherein the runtime metadata comprises content contained in query results.

13. (Canceled)

14. (Previously Presented) The method of claim 10, wherein obtaining metadata associated with the functional module comprises examining a signature validation.

15. (Previously Presented) The method of claim 10, wherein the metadata associated with at least one of the functional modules comprises at least one of: one or

more input parameters required for successful execution of the functional module, one or more output parameters required for successful execution of the functional module, and a security credential required to execute the functional module.

16. (Original) The method of claim 10, wherein at least one of the functional modules analyzes query results.

17. (Original) The method of claim 16, wherein:

the runtime metadata comprises the names of fields in a result set; and

the limited subset of functional modules comprises functional modules requiring data from fields in the result set as inputs.

18. (Original) The method of claim 10, wherein the runtime metadata comprises information related to a query building session.

19. (Original) The method of claim 18, wherein:

the information related to the query building session comprises a specified focus of the query; and

identifying a limited subset of the functional modules that will successfully execute comprises identifying functional modules associated with the specified focus.

20. (Previously Presented) A computer readable medium containing a program which, when executed, performs operations for providing a user access to functional modules from within an application, comprising:

assigning metadata requirements to functional modules that operate on data stored in, or functional modules that generate results that are stored in, a database, wherein the assigned metadata requirements specify conditions required for successful execution of the functional module;

collecting runtime metadata relating to a query session;

obtaining a list of functional modules accessible from within the application

identifying a limited subset of functional modules that will successfully execute, by comparing the collected runtime metadata with the assigned metadata requirements; and

providing an interface presenting the user with the identified limited subset of functional modules that will successfully execute.

21. (Original) The computer readable medium of claim 20 wherein the application is a query building application.

22-26. (Canceled)

27. (Previously Presented) A data processing system for providing a user access to functional modules from within an application comprising:

a data repository;

a plurality of functional modules, each having associated metadata requirements that specify conditions required for successful execution of the functional modules;

an application from which the functional modules are accessible, wherein the application is configured to collect runtime metadata after the composition of a query and present to a user a limited subset of the functional modules that will successfully execute, as determined by the application based on the collected runtime metadata and the metadata requirements associated with the functional modules.

28. (Original) The data processing system of claim 27, wherein the data repository comprises XML data structures used to store runtime metadata.

29. (Original) The data processing system of claim 27, wherein the data repository comprises relational database tables used to store runtime metadata.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.